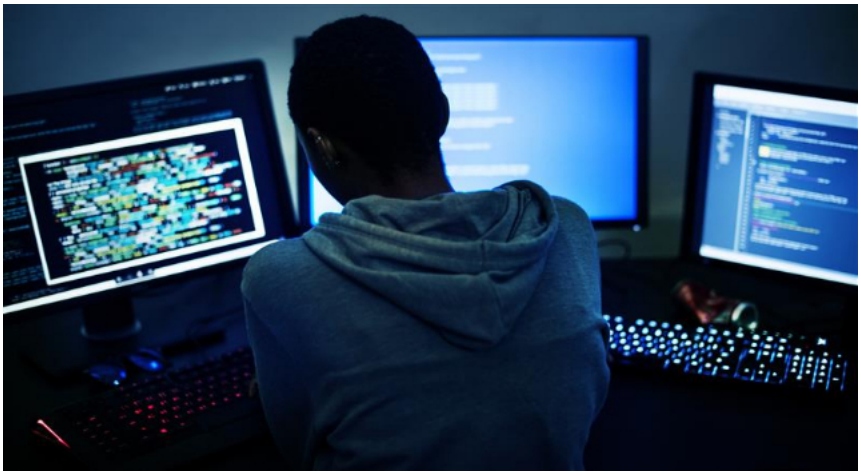
Weaponizing Open Source Software for Targeted Attacks

trendmicro.com/en_us/research/20/k/weaponizing-open-source-software-for-targeted-attacks.html

November 20, 2020



Trojanized open-source software is tricky to spot. This is because it takes on the façade of legitimate, non-malicious software, making it especially stealthy and useful for targeted attacks. However, a closer investigation can reveal suspicious behavior that exposes their malicious intent.

How are open-source software trojanized? How can we detect them? To answer these questions, let us walk through a recent investigation we conducted that involved this file type.

The investigation

In our analysis of an incident, we found a file named notepad.exe sticking like a sore thumb. As most already know, Notepad is a well-known legitimate application. However, some malware authors camouflage malicious files by using the name of legitimate software such as notepad.exe to evade detection.

Figure 1. Telemetry data showing the suspicious notepad.exe file

The notepad.exe file was dropped through ntoskrnl.exe, short for Windows NT operating system kernel executable. This was done by either exploiting ntoskrnl.exe or via network shares. Based on the telemetry data we obtained, it's most probably the latter. Performing Root Cause Analysis (RCA) shows that this malicious notepad.exe file has done suspicious actions by calling the following tools:

Executable File	Function
ipconfig.exe	gets Windows IP Configuration

net.exe	• enumerates local and global groups in the domain • lists the settings of server and workstation service • identifies all the shares on the local machine and in the domain • names user local and domain user accounts
reg.exe	dumps import registry keys/entries to a file
systeminfo.exe	gathers operating system configuration information for a local or remote machine, including service pack levels
tasklist.exe	gets a list of currently running processes on either a local or remote machine

Table 1. Executable file names and functions

The notepad.exe file's link to these processes and their functions indicates that the file is a typical backdoor that gets commands from a malicious remote user. However, something caught our attention. The details listed in the file properties of notepad.exe show this:

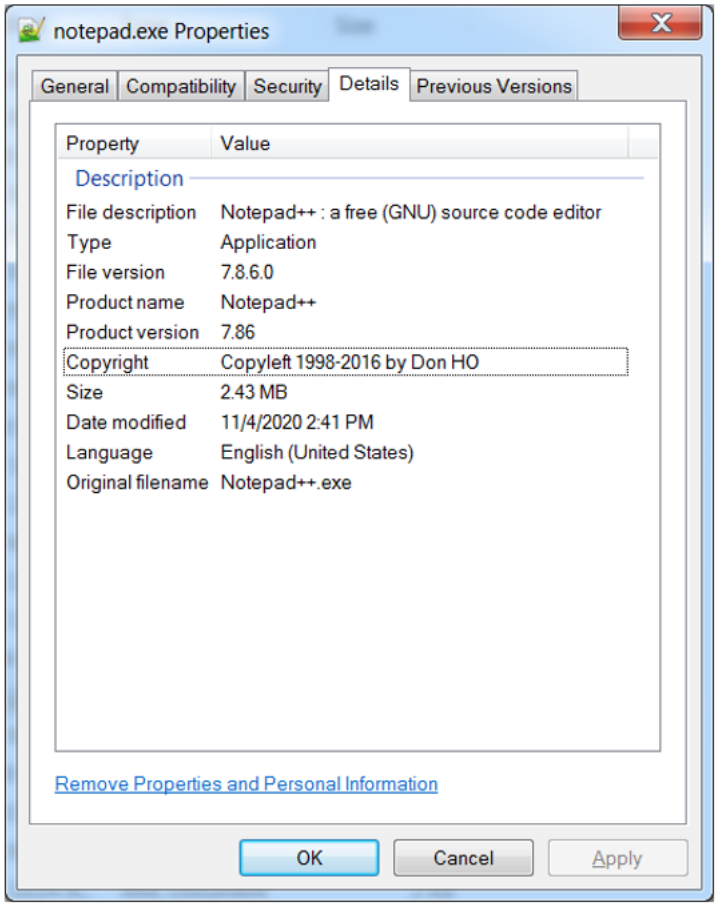


Figure 2. Notepad.exe properties

The file description, product name, and original filename mention Notepad++, an open-source software used as a source code editor. It can also be observed that some of the file's details are dubious. For example, Notepad++ files are usually named as "notepad++.exe" and not "notepad.exe" as seen in this sample. The version — v7.8.6 released in April — is also old; as of writing, the latest version is v7.9.1, released in early November.

Executing the file in question shows this:

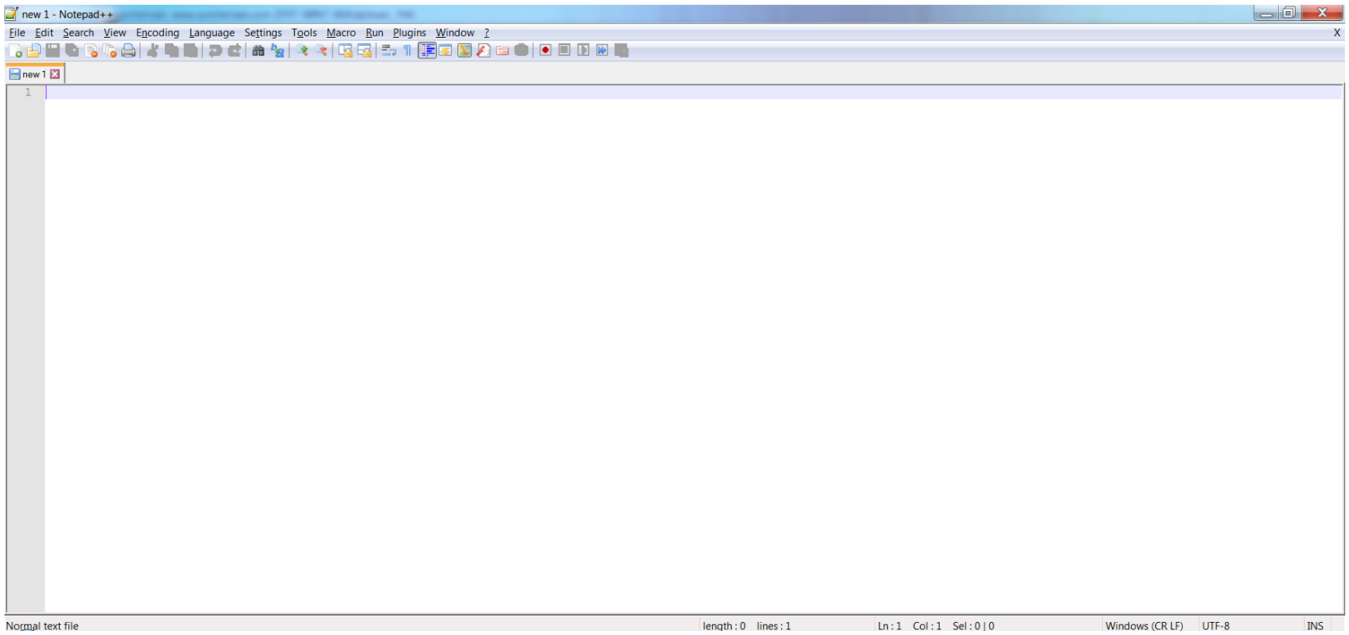


Figure 3. Executed notepad.exe file

The user interface of the file looks and functions convincingly like a typical legitimate Notepad++ file. An initial look reveals nothing suspicious. But in terms of behavior, we discovered that the sample does something that a non-malicious Notepad++ won't do: it searches for a file named config.dat located in c:\windows\debug folder. This behavior is notable as the said file figures in the analysis of the sample's code.

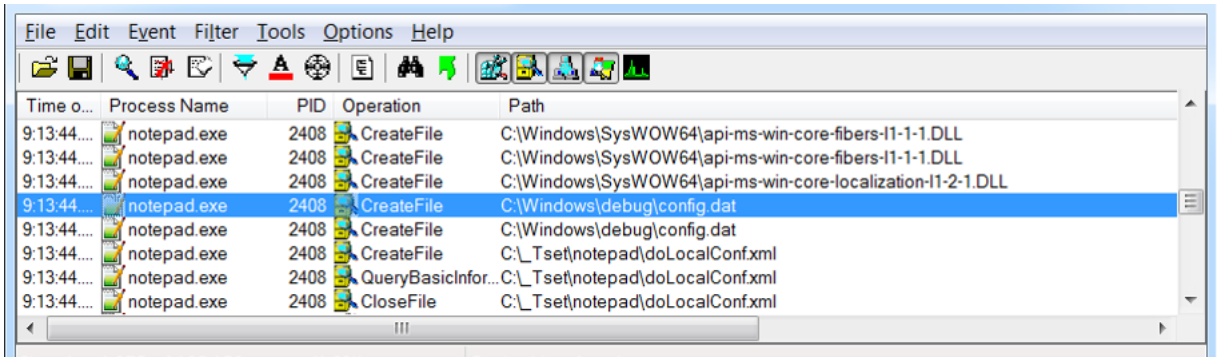


Figure 4. Searching for config.dat file

Code analysis

Decompiling the code of this malicious Notepad++ file shows the following code:

```
v145 = sub_4C59B4(&lpString1);
v134 = sub_4C5C39(110, &lpString1, &v121);
v135 = sub_4C5C39(99, &lpString1, &v121);
v136 = sub_4C5C39(112, &lpString1, &v121);
v137 = sub_4C5C39(120, &lpString1, &v139);
v138 = sub_4C5C39(121, &lpString1, v140);
DecryptedCode = 0;
hFile_config_dat = CreateFileA("c:\\windows\\debug\\config.dat", 0x80000000, 0, 0, 3u, 0x80u, 0);
hFile = hFile_config_dat;
if ( hFile_config_dat != (HANDLE)-1 )
{
  v57 = GetFileSize(hFile_config_dat, 0);
  nNumberOfBytesToRead = v57;
  if ( v57 != -1 )
  {
    DecryptedCode = (unsigned int)VirtualAlloc(0, v57, 0x3000u, 0x40u);
    if ( DecryptedCode )
    {
      NumberOfBytesRead = 0;
      if ( ReadFile(hFile, (LPVOID)DecryptedCode, nNumberOfBytesToRead, &NumberOfBytesRead, 0) )
      {
        v59 = 0;
        for ( i2 = nNumberOfBytesToRead; v59 < i2; ++v59 )
          *(_BYTE *)(v59 + DecryptedCode) = (((*(_BYTE *)(v59 + DecryptedCode) + 33) ^ 0x80) + 3) ^ 0x80;
      }
    }
  }
  CloseHandle(hFile);
}
DeleteFileA("c:\\windows\\debug\\config.dat");
if ( v112 )
  MessageBoxW(
    0,
    L"Usage :\r\rnotepad++ [--help] [-multiInst] [-noPlugin] [-lLanguage] [-LlangCode] [-nLineNumber] [-cColumnNumber] ["
     "-pPosition] [-xLeftPos] [-yTopPos] [-nosession] [-notabbar] [-ro] [-systemtray] [-loadingTime] [-alwaysOnTop] [-o"
     "penSession] [-r] [-qnEasterEggName | -qtText | -qfCntentFileName] [-qSpeed1|2|3] [-quickPrint] [-openFoldersAsWor"
     "kspace] [filePath]\r\r--help : This help message\r-multiInst : Launch another Notepad++ instance\r-noPlugin : Lau"
     "nch Notepad++ without loading any plugin\r-l : Open file or display ghost typing with syntax highlighting of choi"
```

Figure 5. Code snippet of the malicious Notepad++ file

The code snippet taken from a typical non-malicious Notepad++ file is shown below:

```
v104 = sub_5083C0(&Src);
v88 = sub_5086D0(110, &Src, &v72);
v89 = sub_5086D0(99, &Src, &v72);
v90 = sub_5086D0(112, &Src, &v72);
v91 = sub_5086D0(120, &Src, &v93);
v92 = sub_5086D0(121, &Src, &v94);
if ( v66 )
  MessageBoxW(
    0,
    L"Usage :\r\rnotepad++ [--help] [-multiInst] [-noPlugin] [-lLanguage] [-LlangCode] [-nLineNumber] [-cColumnNumber] ["
     "-pPosition] [-xLeftPos] [-yTopPos] [-nosession] [-notabbar] [-ro] [-systemtray] [-loadingTime] [-alwaysOnTop] [-o"
     "penSession] [-r] [-qnEasterEggName | -qtText | -qfCntentFileName] [-qSpeed1|2|3] [-quickPrint] [-openFoldersAsWor"
     "kspace] [filePath]\r\r--help : This help message\r-multiInst : Launch another Notepad++ instance\r-noPlugin : Lau"
     "nch Notepad++ without loading any plugin\r-l : Open file or display ghost typing with syntax highlighting of choi"
     "ce\r-L : Apply indicated localization, langCode is browser language code\r-n : Scroll to indicated line on filePa"
     "th\r-c : Scroll to indicated column on filePath\r-p : Scroll to indicated position on filePath\r-x : Move Notepad"
     "++ to indicated left side position on the screen\r-y : Move Notepad++ to indicated top position on the screen\r-n"
     "osession : Launch Notepad++ without previous session\r-notabbar : Launch Notepad++ without tabbar\r-ro : Make the"
     " filePath read only\r-systemtray : Launch Notepad++ directly in system tray\r-loadingTime : Display Notepad++ loa"
     "ding time\r-alwaysOnTop : Make Notepad++ always on top\r-openSession : Open a session. filePath must be a session"
     " file\r-r : Open files recursively. This argument will be ignored\r     if filePath contain no wildcard character"
     "\r-qn : Launch ghost typing to display easter egg via its name\r-qt : Launch ghost typing to display a text via t"
     "he given text\r-qf : Launch ghost typing to display a file content via the file path\r-qSpeed : Ghost typing spee"
     "d. Value from 1 to 3 for slow, fast and fastest\r-quickPrint : Print the file given as argument then quit Notepad"
     "++\r-openFoldersAsWorkspace: open filePath of folder(s) as workspace\rfilePath : file or folder name to open (abs"
     "olute or relative path name)\r",
    L"Notepad++ Command Argument Help",
    0);
v11 = sub_40BAA0(v54, v55);
v12 = (void **)v98[0];
if ( v100 < 8 )
  v12 = v98;
if ( sub_406240(v12, v99, &::Src, 0) && (void **)(v11 + 144000) != v98 )
  sub_406AC0(v98, 0, -1);
sub_4B78E0(v54, v55);
v72 = *(_BYTE *)(v11 + 680);
v13 = v70;
if ( v65 || v70 )
```

Figure 6. Code snippet of a typical non-malicious Notepad++ file

These code snippets bear many similarities. However, the malicious Notepad++ file has additional code that loads an encrypted blob file (config.dat) that decrypts the code and executes it in the memory so it can perform its backdoor routines. This reminds us of some older malware types like PLUGX.

We observed two instances using the same loader but delivering different payloads. One of the payloads is detected as TrojanSpy.Win32.LAZAGNE.B, while the other is detected as Ransom.Win32.EXX.YAAK-B (Defray ransomware). Further investigation also revealed other blob files with the same loader which lead to different payloads.

We suspect that the file in this incident got into the organization through a targeted watering hole attack. After the initial machine was infected, propagating the malicious notepad++ and config.dat via admin shares would be easy. The notepad.exe file that we investigated came from malicious sources and are not associated with official distributing sites of Notepad and Notepad++.exe.

Weaponizing open-source software

Due to its uncanny resemblance to a legitimate Notepad++ file, the analyzed sample can be easily mistaken as a non-malicious file, especially by employees with limited technical know-how. Threat actors achieved this disguise by trojanizing open-source software. Notepad++'s source code is available publicly; thus, anyone (including malware authors) can access it.

Threat actors can look for open source code of widely-used software and trojanize it by adding malicious code that can perform functions like loading an encrypted blob file. This means that most of the resulting file's binary code is outright non-malicious, and the malicious code simply loads a file, an activity that does not seem to be too suspicious. Additionally, encrypted blob files don't have file headers. These make it difficult for antimalware solutions, including AI/ML-based ones and those that only focus on a single protection layer, to detect. To block these types of threats, solutions that grant visibility across layers would be helpful, as security teams can use them to correlate data and behavior within the environment.

Recommendations

Users should only download files, applications, and software (such as open-source software) from trusted and legitimate sources to avoid these types of threats. For example, Notepad++ users can download relevant files from their official website. Enterprises can create and disseminate a list of approved downloading sites to their employees. As a further security measure, companies can also require the approval of IT teams before employees can install any software on office devices. For security and IT Teams, it is also strongly recommended to validate the downloaded binary with checksums, as good open source projects maintain checksums of their official released binaries.

We also recommend Trend Micro™ XDR, which collects and correlates data across endpoints, emails, cloud workloads, and networks, providing better context and enabling investigation in one place. This, in turn, allows teams to detect advanced and targeted threats earlier.

Indicators of Compromise

File name	SHA-256	Trend Micro Pattern Detection	Trend Micro Ma Learning Detect
notepad.exe (malicious, non-legitimate file named as such)	bacc02fd23c4f95da0fbc5c490b1278d327fea0878734ea9a55f108ef9f4312e	Trojan.Win32.VATET.SM	BKDR.Win32.TR
config.dat	64ba94000e2815898fb17e93deaa44ac0e1b4c55316af727b908dfe74c3b7ef6	Trojan.Win32.VATET.ENC	N/A
config.dat	33234dc94d926f1fc2831f40e27080739b415d485aa457d14a83617a3996089b	Trojan.Win32.VATET.ENC	N/A
release.exe	09c99e37121722dd45a2c19ff248ecfe2b9f1e082381cc73446e0f4f82e0c468	TrojanSpy.Win32.LAZAGNE.B	Troj.Win32.TRX.
virus2.dll	1c3331b87dc55a8cc491846f2609d6226f66eb372716df349567ed619dd1b731	Ransom.Win32.EXX.YAAK-B	Troj.Win32.TRX.

Other related hashes:

SHA-256	Trend Micro Patten Detection	Trend Micro Machine Learnin Detection
0b42bf15b77cfe9f9e693f2776691647e78a91be27f5bdb8d1a366be510a773f	Trojan.Win32.VATET.A	Troj.Win32.TRX.XXPE50FFF0
10c4067908181cebb72202d92ff7a054b19ef3aada939bf76178e35be9506525	Trojan.Win32.VATET.A	BKDR.Win32.TRX.XXPE50FFF
19938becb018e3459b49381c7efffabbe44a6450362b769ba85a3f1240b068d0	Trojan.Win32.VATET.A	Troj.Win32.TRX.XXPE50FFF0
2f149a79f721bb78eb956f70183b531fb6a1b233ceb4a3d6385759a0b0c16fd3	Trojan.Win32.VATET.SM	Troj.Win32.TRX.XXPE50FFF0
37e8d3ae4c34441b30098d7711df8ef0bcc12c395f265106b825221744b956bc	Trojan.Win32.VATET.A	BKDR.Win32.TRX.XXPE50FFF
382d9bf5da142d44de5fda544de4fffe2915a3ffc67964b993f3c051aa8c2989	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF

42f5f1b08c9cee876bafdb6dc4188e8e29d26a07951e1083e08e2a4b0cb6d0ff	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF (GENERIC: Hit Bad Auto Shield
4421720e0321ac8b3820f8178eb8a5ff684388438b62c85f93df9743a1d9fdb9	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF
4fb94877cc150f591e5b61dc5641f33e93e67ae1912c2e122e7ef2a236046f1a	Trojan.Win32.VATET.A	BKDR.Win32.TRX.XXPE50FFF
52d3ebe824ad60a939d64e73336e790884e3674b2d22dbe6e3c6b22061124161	Trojan.Win32.VATET.SM	n/a
57eea67e3eebde707c3fb3473a858e7f895ae12aad37cc664f9c0512c0382e6a	Trojan.Win32.VATET.SM	Troj.Win32.TRX.XXPE50FFF03
6ac07424e5c9b87d76645aa041772ac8af12e30dc670be8adf1cf9f48e32944b	Backdoor.Win32.VATET.CFH	BKDR.Win32.TRX.XXPE50FFF
bacc02fd23c4f95da0fbc5c490b1278d327fea0878734ea9a55f108ef9f4312e	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF
ea6c3b993d830319b08871945cf2726dd6d8e62e8fed8fc42bcb053c38c78748	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF
e5ce1c1b69bd12640c604971be311f9544adb3797df15199bd754d3aefe0a955	Trojan.Win32.VATET.A	BKDR.Win32.TRX.XXPE50FFF
ef7e21d874a387f07a9f74f01f2779a280ff06dff3dae0d41906d21e02f9c975	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF
f0a25444cf58b61ff6cdd86ff1cfa53a51ad426817a33bd0e098f4f0ff286f22	Trojan.Win32.VATET.SM	BKDR.Win32.TRX.XXPE50FFF

Cyber Threats

How are open-source software trojanized? How can we detect them? To answer these questions, let us walk through a recent investigation we conducted that involved this file type.

By: Abraham Camba, Bren Matthew Ebriega, Gilbert Sison November 20, 2020 Read time: (words)

Content added to Folio